Help me write a poem: Instruction Tuning as a Vehicle for Collaborative Poetry Writing

Tuhin Chakrabarty[1]* **Vishakh Padmakumar**[2]* **He He** [2,3]
[1]Department of Computer Science, Columbia University
[2]Center for Data Science, New York University
[3]Department of Computer Science, New York University
tuhin.chakr@cs.columbia.edu, vishakh@nyu.edu, hhe@nyu.edu

Abstract

Recent work in training large language models (LLMs) to follow natural language instructions has opened up exciting opportunities for natural language interface design. Building on the prior success of LLMs in the realm of computer-assisted creativity, we aim to study if LLMs can improve the quality of user-generated content through collaboration. We present *CoPoet*, a collaborative poetry writing system. In contrast to auto-completing a user's text, CoPoet is controlled by user instructions that specify the attributes of the desired text, such as *Write a sentence about 'love'* or *Write a sentence ending in 'fly'*. The core component of our system is a language model fine-tuned on a diverse collection of instructions for poetry writing. Our model is not only competitive with publicly available LLMs trained on instructions (InstructGPT), but is also capable of satisfying unseen compositional instructions. A study with 15 qualified crowdworkers shows that users successfully write poems with CoPoet on diverse topics ranging from *Monarchy* to *Climate change*. Further, the collaboratively written poems are preferred by third-party evaluators over those written without the system.[1]

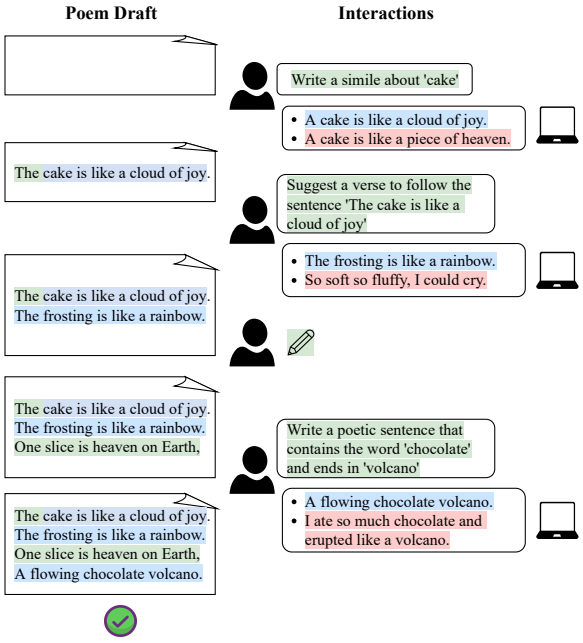

Figure 1: A collaborative poem entitled '*Decadence*', written with CoPoet assistance. Green text was written directly by the human, who interacts with CoPoet using instructions. CoPoet offers multiple suggestions which the user can accept or reject. The user wrote a four line poem before indicating completion of the task.

1 Introduction

Advancements in large language models (LLMs) have made remarkable progress towards generating coherent text in a wide variety of domains. This has spurred increasing interest in computer-assisted creativity (See et al., 2019; Elkins and Chun, 2020; Ramesh et al., 2022; Branwen, 2020) such as building co-creative assistants for writing stories, poems, and argumentative essays (Lee et al., 2022; Swanson et al., 2021; Uthus et al., 2019; Donahue et al., 2020; Padmakumar and He, 2022; Du et al., 2022). The adoption of these technologies hinges on their ability to provide appropriate suggestions while being easy to interact with. However, there has been limited research on the effectiveness of such collaboration, e.g., whether the assistant understands user intents and whether collaboration improves the final outcome.

In this paper, *we aim to understand the collaboration capabilities of LLMs through a case study of collaborative poetry writing.* Writing a poem is often a challenging task because it is both open-ended and highly constrained. Unlike stories or other argumentative texts, in order to write a poem we need creative content that satisfies various long- and short-range form constraints such as rhyme, meter, and sound, which poses a significant challenge for end-to-end poem generation systems (Ghazvinine-

*Both Authors Contributed Equally

[1]Our code, preprocessed data, models, and the interaction logs from our user study are available at https://github.com/vishakhpk/creative-instructions

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 6848–6863
December 7-11, 2022 ©2022 Association for Computational Linguistics

jad et al., 2016; Tian and Peng, 2022; Van de Cruys, 2020; Ormazabal et al., 2022). While LLMs sometimes struggle with long-range coherence, they are good at providing variations of text that satisfy local constraints. This makes them great partners to humans in poem writing, where humans focus on the long-range writing plan and the machine implements the ideas locally.

Effective collaboration in co-creative writing is challenging as it requires the model to understand user intention. For example, as shown in Figure 1, a user may have a rough plan around two related concepts such as chocolate and volcano, and want the model to suggest a verse that contains *chocolate* and ends with *volcano*; or they may be looking for a verse that rhymes with a specific word (*rainbow*) to satisfy the constraints. An auto-completion interface is not able to anticipate such user needs and provide targeted suggestions. To enable richer interaction, we rely on instructional prompts (Wang et al., 2022; Sanh et al., 2021; Mishra et al., 2022; Mishra and Nouri, 2022) that act as a natural language interface between the user and the assistant.

Specifically, we present *CoPoet*, a collaborative poem writing system with a natural language interface. During a writing session, the user can iteratively request suggestions through natural language instructions such as *Write a simile about 'cake'*, and edit their draft based on the suggestions (Figure 1). To build *CoPoet*, we finetune a pretrained sequence-to-sequence model on a parallel corpus of instruction-output pairs. We obtain the outputs from publicly available datasets of creative text and synthesize the corresponding instructions by rules, including both lexical and rhyming constraints as well as requests on rhetorical devices.

To understand how well the model follows instructions, we test it on instructions with varying levels of difficulty, from those seen during training to unseen compositional instructions that contain multiple constraints. Both automatic and human evaluation show that our finetuned model satisfies the constraints 86% of the time, 10% better than a much larger 175B version of InstructGPT (Brown et al., 2020). On unseen compositional instructions, our best model satisfies them 77.6% of the time, outperforming InstructGPT by a margin of 28%.

To understand its collaboration capabilities, we run a user study on Amazon Mechanical Turk (AMT) where *CoPoet* assists expert crowd workers (recruited through a qualification test) in writing poems (Section 4). We observe that the recruited users are able to write coherent and creative poems on diverse topics ranging from *Glass Ceiling* to *Climate Change*. About 70% of model suggested text is retained in the final poem and users give *CoPoet* a rating of 4.3 out of 5 on both the suggestion quality and the overall helpfulness. Further, a separate group of annotators on AMT prefers the collaboratively written poems more often than those written without *CoPoet* assistance. In particular, we find model assistance improves rhyming and vocabulary diversity of the poems.

Subject	Write a poetic sentence about 'sun'
	O crimson sun, your warming draft's pulsation.
End	Write a poetic sentence ending in 'glory'
	Am I exalted here unto that glory.
Rhyme	Write a poetic sentence that ends in a word which rhymes with 'replace'
	Diminishing for me, with delicate grace.
Next Sentence	Write a next sentence in a poem given the previous sentence ''The only thing I know' 'for sure'
	Is that love is hard and can be obscure.
Metaphor	Write a metaphor about 'brain'
	My brain is a tangled mess of circuits.
Simile	Write a simile about 'making someone feel desired'
	I want to make you feel like a flower near a hummingbird
Onomatopoeia	Write a poetic sentence about 'bottles' showcasing onomatopoeia
	The stamping of feet and the ring of bottles.
Subject + End	Write a poetic sentence about 'tears' and ending in 'wives'
	Awash in the tears of soldier's wives.

Table 1: Natural language instructions for poem writing paired with example outputs. Each instruction consists of a template and an argument.

2 Data

To train a model to follow instructions, we need `<instruction, poem_line>` pairs where the text satisfies the instruction. The key challenge to building such a model is the lack of parallel data, so we collect our own dataset of creative writing instructions from publicly available poem corpora or relevant subreddits from Reddit (Table 7).

Based on some initial feedback from professional poets, we decided to include 3 major types of instructions: 1) *Continuation* based in-

structions that suggest content when writers are blocked/clueless on how to proceed; 2) Instructions on *Lexical Constraints* to enable greater control of poetic form such as rhyme, sound, and meter. These are instructions that force language models to obey specific choices such as generating a line that contains a specific *topic*, *start word*, *end word* or a sentence with a particular *rhyme*; 3) Instructions on *Rhetorical devices* that are mostly used for introducing embellishments and imagery in a poem such as *metaphor, similes, and onomatopoeia*.

Table 1 shows the primary instructions used to train our models. These instructions are crafted by the authors of the paper, who convert every poem line to an `<instruction, poem_line>` pair using rules.

Each instruction consists of a *template* (unique to the instruction type) and one or more *arguments*, as can be seen in Table 1. Given a poem line in the corpus, we reverse-engineer the instruction by picking a template and extracting the arguments from the poem line. For continuation instructions, we use the previous context as the argument. For instructions on lexical constraints, we extract noun phrases and start/end words as arguments using NLTK for tokenization. To construct instructions on rhymes, we use the CMU dictionary to find rhyming words.[2] We describe more details in Appendix A on how we create instructions for each particular type.

To allow models to adapt to linguistic variations of the instruction templates, we also include paraphrases of the instruction templates, e.g., instead of "Write" we also use"Generate", or instead of "Write a sentence about" we use "Write a sentence that contains the word" or "Write a sentence that includes the word". In total, our dataset consists of 873,574 `<instruction, poem_line>` pairs which we randomly split into 808,180 train and 65,394 held-out validation examples.[3] We evaluate performance on three test sets of hand-crafted instructions of varying difficulty (Section 3.2).

3 How Well Do LLMs Follow Instructions?

In this section, we first describe our models and baselines, followed by the evaluation results using both automatic metrics (Section 3.3) and human evaluation (Section 3.4).

3.1 Experiment Setup

Model Details We finetune the pretrained T5 (Raffel et al., 2020) and T0 (Sanh et al., 2021) models from HuggingFace (Wolf et al., 2019) on the collected data (Section 2) to produce the output given the instruction using cross-entropy loss. We report results on finetuned T5-3B, T5-11B and T0-3B models, which are henceforth referred to as T5-3B-poem, T5-11B-poem, and T0-3B-poem. We select the hyperparameters by the validation loss: for T5-11B-poem, we use the Adam optimizer with a learning rate of $1e^{-4}$; for T5-3B-poem and T0-3B-poem, we use the Adafactor optimizer with a learning rate of $1e^{-3}$. Each model is trained for 3 epochs with early stopping based on validation loss. We finetune all models on an A100 GPU and use Deepspeed (Rasley et al., 2020) integration for the 11B model. During finetuning, we restrict the maximum sequence length of both the source and the target to 64 tokens (via truncation).[4] At inference time, we generate output sequences using top-k sampling with $k = 5$ and a temperature of 0.7 per recommendations from earlier work in open-ended creative text generation (Fan et al., 2018; Holtzman et al., 2020; Padmakumar and He, 2022).

Baselines We compare our finetuned models with two other models: (i) the T0pp model (Sanh et al., 2021), trained on instruction-based prompts from 49 datasets;[5] and (ii) the 175B davinci variant of InstructGPT (Ouyang et al., 2022) that is trained on human-written instructions on diverse tasks in a human-in-the-loop fashion. Given an instruction, we generate text directly (i.e. zero-shot) from T0pp using top-k sampling (Fan et al., 2018). For InstructGPT, we evaluate on both zero-shot and few-shot settings. For zero-shot, the prompt consists of only the instruction. For few-shot, the prompt consists of 26 `<instruction, poem_line>` pairs from our training data (selected to cover all the instruction templates), followed by the test instruction.[6] We use the OpenAI API with a temperature of 0.7, no frequency penalty, and a maximum sequence length of 64 to match our setting.

[2] https://pypi.org/project/pronouncing/

[3] Our dataset is publicly available at https://github.com/vishakhpk/creative-instructions.

[4] The length limit is chosen to avoid memory explosion. It has minimal impact on model performance since most verses are shorter.

[5] These include question-answering, summarization, structure-to-text generation, sentiment and topic classification tasks but no explicit creative writing tasks.

[6] The exact prompt can be found in our code repository.

3.2 Test Sets

While our training instructions cover many templates and topics, user instructions may deviate from the training distribution during interaction. To evaluate the generalization capabilities of the models, we identify three settings with increasing difficulty based on whether the instruction templates or arguments are seen during training.

Known Instruction Templates with Known Arguments (KIKA) The simplest setting requires the model to generalize to novel combinations of the templates and arguments. Specifically, we create instructions where both the templates and the arguments are seen in the training set, although each specific combination is unseen (i.e. the training and test sets have no overlapping instructions).

Known Instruction Templates with Unknown Arguments (KIUA) To handle novel concepts from users, the model must generalize to unseen arguments, which may include new entities or phrases. For example, it might be easier for a model to write a poetic sentence about a known argument such as *beauty*, but difficult to write about an unknown argument *beauty without virtue*. For this set, we include instructions where the instruction templates are seen during training but the corresponding arguments are unseen.

Unknown Compositional Instruction Templates One of the main benefits of natural language instructions is that they can be easily composed in new ways to cover various user intentions. This is particularly useful in creative writing because it enables users to request text from the model with multiple constraints. Therefore, we also test whether the model understands compositional instructions using two templates, as seen in Table 2. Our model is exposed to a *single* compositional template during training: *Subject+End*. For this test set, we create a variety of unseen compositions.

In total, we create 242 test examples (82 KIKA, 82 KIUA, 78 compositional) by selecting instructions according to the above criteria, followed by manual verification.

3.3 Automatic Evaluation

We evaluate how well the models satisfy constraints specified in the instructions on each of the test sets (Section 3.2). We report the success rate of satisfying the instructions where the success condition for each instruction type is listed in Table 3.[7]

Start +End	Write a poetic sentence that starts with the word 'Maybe' and ending in 'void'
	Maybe one day, you will find me in the void
Subject +Rhyme	Write a poetic sentence that contains the word 'breaks' and ending in a word which rhymes with 'bound'
	She cracks and breaks and hits the ground.
Next Sentence +End	Write a next sentence in a poetry given the previous sentence 'Every once a while I lower the blinds' and ending in 'play'
	Waiting for someone to call me out to play
Metaphor +End	Write a metaphor that includes the word 'film' and ending in 'thought'
	A film is a petrified fountain of thought.

Table 2: Examples of compositional natural language instructions for creative tasks paired with their respective outputs from our test sets.

Instruction Type	**Success Condition**
Rhyme	Last word of the model generation rhymes with the desired subject using the CMU Pronouncing Dictionary
Haiku	Model generation contains 15–19 syllables and contains the desired subject
Simile / Metaphor	Model generation contains the desired subject as well as a comparator
Start / End	First/last word of the model generation matches the desired subject
Subject	Model generation contains the desired subject in the instruction

Table 3: Success conditions for different instruction templates.

Finetuned Models Have Strong In-Domain Performance but Drop on Out-of-Domain Data Figure 2 shows the average success rate and standard deviations of each model on the three test sets across 5 model inferences to account for variance in top-k sampling. On both KIKA and KIUA, T5-11B-poem has the highest average success rate. T5-3B-poem and T0-3B-poem outperform the few-shot and zero-shot baselines on both test sets. However, these finetuned models suffer a big drop in performance from KIKA to KIUA—T5-11B-poem

[7]Prior work on instruction tuning reports metrics such as BLEU score for generation tasks (Sanh et al., 2021; Wei et al., 2021) and these are unsuitable for our poetry writing instructions, thus we define custom success conditions.

suffers a relative drop of 51.09% from a 73.2% success rate on KIKA to a 35.8% rate on KIUA. In contrast, the few-shot InstructGPT baseline only suffers a relative drop of 30.4% from a success rate of 46.6% on KIKA to 32.4% on KIUA. This result is consistent with prior findings that task-specific finetuning may destroy pretrained representation which leads to degrading performance on other non-finetuning tasks (Aribandi et al., 2021; Padmakumar et al., 2022). Without finetuning, in-domain examples are still helpful though: on all test sets, the InstructGPT few-shot baseline outperforms the corresponding zero-shot baseline along with a reduction in variance across runs.

Larger Models Compose Instructions Better On compositional instructions, we find that T5-11B-poem has the best average performance. In addition, there is a clear performance gap between the 11B and 3B models, showing the importance of model scale for composition, similar to recent observations of *emergent* abilities in LLMs (Wei et al., 2022). We also find that few-shot InstructGPT outperforms T5-3B-poem and T0-3B-poem despite having no compositional instructions in the prompt. This indicates that smaller models, when finetuned on instructions, tend to overfit to templates seen during training, which hurts their generalization capability, as also reported in Wei et al. (2021).

3.4 Human Evaluation

Since our automatic metrics are not always accurate in measuring if an instruction is satisfied, we also perform human evaluation by having crowd workers manually check if model generations satisfy the instruction constraints. Given the automatic evaluation results in Section 3.3, we compare our best finetuned model, T5-11B-poem, against the top performing baseline, few-shot InstructGPT. Specifically, we conduct pairwise comparison: each annotator is shown an instruction and generations from both models.[8] They are asked to rate the fluency, accuracy, and creativity of the generation by answering the following questions:

- Rate the fluency of each verse on a scale of 1–5.
- Does each verse adequately satisfy the instruction? (Yes/No)
- Which of the two verses is *more creative/interesting* while being *coherent* and *satisfying the instruction*?

[8]We sample 5 generations from each model and select the best one using the criteria in Table 3. If multiple candidates are evaluated as success, we randomly sample one.

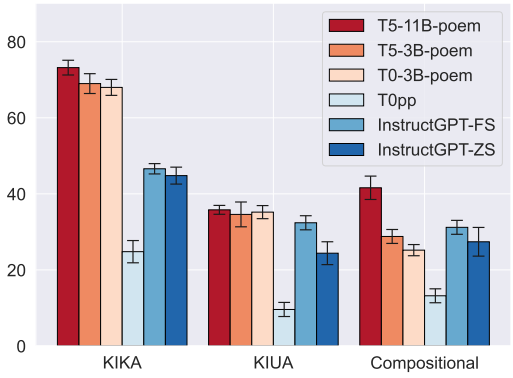

Figure 2: Automatic evaluation of models on KIKA, KIUA and Compositional test sets. The y axis is the percentage of instructions that each model successfully satisfies as determined by the criteria in Table 3. We report results on T5-11B-poem, T5-3B-poem and T0-3B-poem along with the baselines—zero-shot T0pp (Sanh et al., 2021) and zero-shot (ZS)/few-shot (FS) InstructGPT (da-vinci) (Ouyang et al., 2022). Each bar shows the average success rate of 5 model inferences along with the standard deviation. On average, T5-11B-poem achieves the highest success rate and InstructGPT is a strong few-shot baseline that obtains comparable results on KIUA.

The first two questions evaluate the quality of each verse against the instruction individually. In addition to satisfying the constraints in the instruction with fluent text, we want the model to provide novel suggestions that are helpful for creative writing. Thus we also ask the annotators to compare the two verses and provide a subjective judgement on which one is more creative. We collect three annotations for each question and use the majority vote as the final judgement.

T5-11B-poem Satisfies Instructions Better than Few-Shot InstructGPT Table 4 shows the human evaluation results on all three test sets. We find that, on average, model generations from T5-11B-poem satisfy the given instructions better on all three test sets, while InstructGPT is rated to be more fluent consistently. We find that gap in satisfying instructions is largest on the compositional test set—T5-11B-poem accurately answers 77.6% of compositional instructions while InstructGPT only manages 55.2%. Annotators also reported that verses from T5-11B-poem were marginally more creative/interesting than InstructGPT on KIKA and KIUA test sets and less so on the Compositional

		T5-11B-poem	GPT3-FS
KIKA (82)	Success%	**86.2**	76.9
	Fluency	0.739	**0.794**
	Creative	**53.8**	46.2
KIUA (82)	Success%	**92.5**	86.5
	Fluency	0.773	**0.781**
	Creative	**56.7**	43.3
Comp (78)	Success%	**77.6**	55.2
	Fluency	0.697	**0.751**
	Creative	47.7	**52.3**

Table 4: Human evaluation of model generations from T5-11B-poem and few-shot InstructGPT3 on different test sets across three metrics: (i) success rate: percentage of instructions satisfied; (ii) fluency: average fluency score on a scale of 5 normalized to $[0, 1]$; (iii) creativity: percentage of generations rated to be more creative / interesting in a pairwise comparison.

test set, indicating that the two models may have little difference in creativity.[9]

We observe that InstructGPT is a strong baseline, outperforming T0pp by a large margin on automatic metrics, and satisfying nearly 80% of the instructions in the KIKA and KIUA test sets according to human evaluation. However, a common error case on compositional instructions is that while the model generations almost always contain the arguments mentioned in the instruction, they do not always satisfy the constraints correctly—when asked for a verse that contains the word 'soul' and ends with 'yellow', InstructGPT generated the line "My soul is as yellow as the sun on a summer day" that contains those arguments but not at the specified positions.

Takeaways We observe that on average finetuned models tend to outperform the few-shot baselines on in-domain instructions (Section 3.3). While smaller models (T5-3B-poem, T0-3B-poem) have worse performance on out-of-domain instructions, finetuned models at scale (T5-11B-poem) generalizes to compositional instructions effectively, even outperforming InstructGPT (Section 3.4). The flexibility of composing instructions makes the model more suitable as a collaborator for a human user; hence we use T5-11B-poem as the assistant for our subsequent collaborative experiments.

[9]The first two questions are less subjective than the third question. Users unanimously agreed 52.2% of the time on whether model generations satisfied instructions and only 37.3% on which output is more creative.

4 CoPoet: Collaborative Poem Writing

Our results in Section 3.4 demonstrate *CoPoet*'s ability to satisfy the constraints specified in the instructions. This presents us with an opportunity to test the model's capability in collaborative writing tasks. We design our user study (Figure 3) to answer the following two main research questions:

- **RQ1:** Can users write poems on any topic of their choice by collaborating with CoPoet?
- **RQ2:** Does CoPoet help users write *better* poems compared to when they write alone?

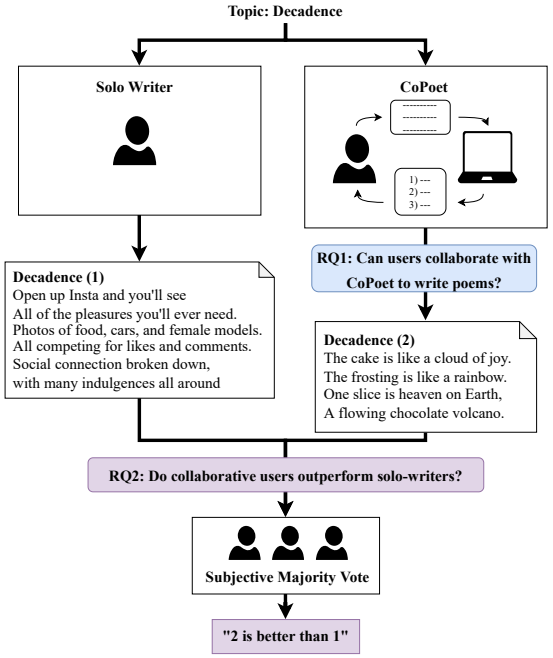

Figure 3: CoPoet user study. We study if users can effectively collaborate with CoPoet to write poems (RQ1) and whether writing with CoPoet produces better poems compared to solo-writers (RQ2).

Interface Design Since we intend to study the task of collaborative poem writing, we develop a user interface for our experiments where users can work on their poem drafts and also query CoPoet for suggestions using written instructions. A screenshot of the interface is provided in Figure 11. In response to each instruction, CoPoet provides 5 suggestions, each in the form of one poem line, to the users. The users can then choose if they wish to incorporate these into their draft. We instruct them to edit the model output when required to ensure the overall coherence of the poem. As seen in Figure 11, users are also provided with the list of instruction templates used to train the

model (Section 2). These are intended to communicate to users the instructions that the model is trained to respond to, so that they have an idea of what the model is capable of.[10]

Experiment Setup We first conduct a qualification test on AMT, where we recruit 50 workers to collaboratively write a poem of four lines using CoPoet. We require a user to interact with our system at least four times (i.e., to issue at least four instructions). However, we do not enforce that they use any of the model outputs in response to their instructions—they are free to ignore all model suggestions. The authors of the paper then independently rank these poems in terms of fluency, richness in imagery, and creativity. Finally, 15 crowd-workers passed the qualification test. From now on, we refer to these qualified workers as *experts*.

We then collect 50 distinct poems collaboratively written by our experts using CoPoet, where they are instructed to write a poem on a topic of their choice. In order to compare collaborative writers to solo-writers, we then collect 50 poems on the same titles from expert writers writing without model assistance.[11] Third-party annotators were then shown the title and two poems interpreting it, and instructed to select the one they felt was a 'descriptive interpretation of the title'. To ensure a fair judgement, both the poems were identical in length (4 lines), randomized in order, and without obvious clues in the vocabulary usage. To the best of our knowledge, there was no underlying bias that would make it easy for judges to identify which poems were collaborative and which were written entirely by humans. The full experiment design is shown in Figure 3.

RQ1: Can experts write poems successfully on any topic of their choice by collaborating with CoPoet?

From our user study, we observe that experts are able to collaborate with CoPoet and write poems on diverse topics of their choice, including *Climate Change*, *Hunger*, *Glass Ceiling*, *Decadence* etc. We include more examples in Appendix B. The full list of titles visualized as a word cloud can be found in Figure 10.

[10]We explicitly mention that they can use novel instructions not present in the templates.

[11]We ensure that the same author does not write on the same topic in the two setups.

How do experts use instructions? On average, experts use 7 instructions per poem. Figure 4 shows that experts often prefer contextual instructions, i.e. getting ideas from the model about the *Next Sentence* given what they have written thus far. The *Topic* instruction is also significantly used, which helps them add control. It is encouraging to see humans using a total of 87 compositional instructions, which constitutes almost 24% of the total set of instructions used. Finally, humans also use figurative embellishments such as *Similes* or *Metaphors* suggested by the model.

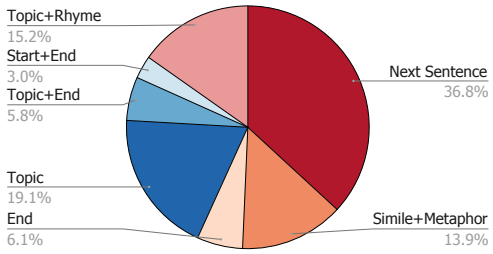

Figure 4: Proportions of the types of instructions used by experts in the poetry writing task.

Do experts find CoPoet helpful as a writing tool? We collect judgments from 15 experts to tease out and characterize the model's contribution. We are interested to know whether the model helped in the writing process by satisfying the instructions, and how well it served the writers' needs. We collect ratings on a Likert scale from 1 (not at all) to 5 (very) on two questions: (i) How accurately does the model follow instructions? (ii) How helpful is the model in the process of writing poetry? We obtain an average score of 4.3 out of 5 on both questions, suggesting that CoPoet is a useful tool for poem writing. Table 8 in Appendix A shows some of the feedback provided by experts, including how they found the system helpful in situations such as *writers' block*, and how specific instructions helped them write better.

What fraction of the poems is written by CoPoet? To quantify the contribution of the model, we compute the proportion of the submitted poems that was taken from the model generations. We calculate this using the Rouge-L recall (Lin and Rey, 2004) score of the poem lines with respect to the model suggestions i.e. what fraction of the poem is found in the generated output of the model. Each verse is greedily matched to a unique model suggestion with the largest overlap. The calculation

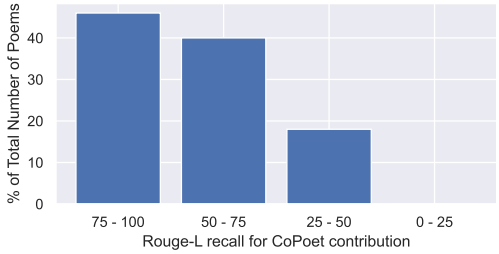

Figure 5: Content overlap between sentences of an individual poem and the corresponding model suggestions calculated using Rouge-L recall. Y axis shows the percentage of poems out of 50 while X axis shows the amount of Copoet contribution in terms of Rouge-L.

	Relevant %	Preferred %
Solo	96	43
Collaborative	98	**57**

Table 5: Human evaluation of 50 poems written by solo-writers vs those written by users with CoPoet. Workers have a slight preference for collaborative poems.

is described in Algorithm 1 in Appendix A. Figure 5 shows that on average 46% of collaborative poems have a Rouge-L recall score greater than 75%, i.e 75% of the content in the collaborative poems are obtained from CoPoet suggestions. Additionally, a further 40% of the collaborative poems have more than half of their content (50-75%) written by the model. This suggests that the majority of the text generated by CoPoet is considered high-quality and usable by the expert users.

RQ2: Can CoPoet help users write better poems compared to when they write alone? To answer the above question, we compare poems written by the set of experts with and without model help, as detailed in Figure 3. We are interested in measuring i) whether poems written are *relevant*, where an relevant poem is defined as *descriptive interpretation* of the title, i.e. it is on-topic. ii) whether poems written by experts with CoPoet are preferred over poems written by solo-writers.

We recruit a total of 49 third-party annotators to compare poems written by experts alone to those written by experts with CoPoet. They are shown one poem each from a solo-writer and a collaborative writer, both in response to the same title, and requested to label each poem on whether it is *relevant*. Additionally, they are asked to choose their preferred poem between the given pair in terms of coherence, overall quality, and style. Each pair of poems is evaluated by 3 distinct annotators. We then aggregate the judgments via *majority voting*. Table 5 shows that both poems written by solo writers and poems written collaboratively are accurate. We are encouraged to see that collaborative poems are preferred more than poems written by solo-writers. These findings suggest that CoPoet is a helpful tool for poetry writing and instructions act as a useful vehicle for co-creative writing using LLMs.

	Preferred %	Not Preferred %
Diversity	63.0	37.0
Rhyme	72.5	27.5
Perplexity	55.0	45.0

Table 6: Analysis of poems preferred by third-party annotators based on (i) rhyme (ii) diversity and (iii) perplexity. Workers' preference is correlated with the presence of rhyming and vocabulary diversity.

Potential Factors for User Preference We acknowledge that there is some degree of subjectivity in the user preferences. To better understand why a certain poem is preferred by crowd-workers, we investigate whether certain factors correlate with their choices. We measure i) *Diversity* (in terms of distinct unigrams) ii) *Presence of Rhyme* (whether there at least one pair of rhyming lines in the poem), and iii) *Perplexity* measured using a pre-trained GPT-2 model for each poem. As can be seen in Table 6, crowd-workers preferred poems that are diverse and have a rhyme scheme 63% and 72.5% of the time. From Figure 4, we know that our experts tend to use the model to express their ideas by eliciting text from the model that contains specific content but is subject to various constraints (*Topic+Rhyme* and the various *End* instructions). Here, we observe that these constraints combined with more diverse vocabulary usage might be contributing to the preference for collaborative poems over solo poems.

5 Related Work

Collaborative Writing The key challenge in collaborative writing is to understand user intent so as to provide timely and useful suggestions. Prior work in story writing (Roemmele and Gordon, 2015; Clark et al., 2018) presented sentence-level continuations at locations specified by a user. Akoury et al. (2020); Lee et al. (2022) took this a step further providing users with a paragraph of text which they could further edit in story writing and ar-

gumentative writing tasks. However, model suggestions of this autocomplete nature were not always helpful, as they often diverged from the user intent (Clark et al., 2018) resulting in only a fraction of generated text being retained (Akoury et al., 2020). Instead of providing a machine-written draft, Padmakumar and He (2022) showed that having the model rewrite text only at locations specified by the user results in more helpful suggestions in the task of creative image captioning.

We focus on the task of collaborative poem writing, which adds an additional challenge as useful suggestions need to satisfy several lexical and form constraints (rhyme, meter, sound). Past work for this task has used retrieval to provide suggestions for substitutions at the word and phrase level (Chen et al., 2014) or verses that follow different styles (Uthus et al., 2022), but these are unable to dynamically generate novel text. In our work, we utilize large language models to generate text that satisfies the various constraints specified by users, with the added benefit that they can spell out these using natural language instructions. Concurrent work has also shown that large language models can help users write scripts and screenplays (Mirowski et al., 2022) and longer stories (Yang et al., 2022) by generating text that incorporates structural context via prompt chaining.

Interaction with Users Recent work in NLP has highlighted the success of generative large language models as interaction interfaces for the task of creative writing. Finetuning models on tasks verbalised as instructions has shown good generalization to unseen instructions (Wei et al., 2021; Sanh et al., 2021; Mishra et al., 2021; Chung et al., 2022). In our work, we focus on a suite of instructions specific to creative writing and additionally evaluate the instruction-tuning setup with real users who iteratively ask for suggestions in natural language.

In addition to fine-tuning models on instructions, large language models are also able to generalize to unseen tasks in a few-shot manner when the task is specified as part of the prompt in natural language (Ouyang et al., 2022). Reif et al. (2022) present a prompting method which performs style transfer in a zero-shot or few-shot manner with only a natural language instruction describing the target style without model fine-tuning or exemplars in the target style. Unlike most of the recent work that prompts large language models to elicit content Coenen et al. (2021) frame collaborative writing as a conversation between a human and a LLM-based dialog system and show how the spontaneous utilities of conversation support a variety of interactions. More recently Mishra and Nouri (2022) propose a prompting strategy where they ask GPT3 specific questions about mood, tone, occasion, or theme for the task of poem generation by using GPT3 as an interaction interface.

6 Conclusion

In this work, we present *CoPoet*, a collaborative poetry writing system that is controlled by user instructions that specify the attributes of any desired text. Our system is built upon a language model fine-tuned on a diverse collection of instructions for poetry writing. Empirical results show that our model is not only competitive with publicly available LLMs trained on instructions (InstructGPT), but also capable of satisfying unseen compositional instructions. A further study with 15 qualified crowd-workers shows that users successfully write poems with CoPoet on diverse topics, which are also preferred by third-party evaluators over poems written by solo-writers. These results show that language models acting as writing assistants are capable of understanding user intents and collaborating with them to improve the final outcome, potentially makes a challenging task such as poem writing more accessible to users.

Going forward we hope to extend our research to more challenging instructions such as converting longer content planning tasks into the instruction tuning setup to assist users with longer story writing. To provide more robust assistance, we also hope to study how to train models that generalize better to completely unseen instructions. Finally, we intend to more holistically study the problem of co-creative writing by not just examining how to train better assistive models but also how to design effective user interfaces for end users.

Limitations

Noisy Training Data We note that our dataset is self-supervised and we use various tools to align lines of poetry from various sources (Table 7) to templated instructions. There might be small errors in the training data such as spelling mistakes in the lines of poetry (an example from our dataset to showcase this is the line "Lay silently *burid* side by side") or slightly convoluted instructions (an

example instruction to highlight this is "Write a poetic sentence that speaks of *nights grow shorter*"). However, each example in the various test sets (Section 3.2) was manually verified by the authors of this work.

Test Set Size Another potential concern is the size of the test sets which were small as each instruction in these was verified by the authors. We provide confidence intervals on the model success rates to mitigate this in Section 3.3.

Design of the User Interface Our user interface presents templates of instructions to users at the point when they query the model for assistance (Figure 11). This primes the users to write instructions similar to the templates—almost all the instructions used by the crowdworkers belonged to the templates provided in the interface (or novel combinations of these). In this work, we did not perform an extensive comparison of different interface designs which could influence the interaction. We further discuss some of the design choices about the user interface in Appendix C.

Ethics Statement

Although we use language models trained on data collected from the Web, which have been shown to have issues with gender bias and abusive language, the inductive bias of our models should limit inadvertent negative impacts. Unlike model variants such as GPT, T5 is a conditional language model, which provides more control of the generated output. Our poetic parallel corpora are unlikely to contain toxic text and are manually inspected by the authors.Technological advances in text generation have had both positive and negative effects. However, interactive, human-in-the-loop generative systems designed especially for literary or poetic text generation such as ours might speed up literary professional's work and make it more enjoyable. We believe that machine generation of poetic text will not lead to the exclusion of human poets. Rather, it will increase human-machine interaction and continue to enhance human performance.

In order to ensure that there are no privacy issues for our train and validation splits, the poems were broken down line by line and shuffled randomly.They do not contain any metadata and as such cannot reproduce the creative value of the original poems.

Appropriate Remuneration of Crowd-workers For all our tasks, we recruit from a pool of crowdworkers in the USA with a minimum of 95% HIT success rate. To complete the human evaluation of model outputs satisfying instructions (Section 3.4), a crowdworker has to read an instruction and two lines in response to it and answer a total of 5 questions. On average, this takes slightly less than two minutes, so we set the payment to \$0.50 per HIT. For the writing tasks (solo and collaborative, Section 4), on average our users take 10 minutes to write a poem, so we set the payment of \$2.50 for each HIT. We also reward writers \$0.50 per poem on submission of poems deemed *relevant* or a relevant interpretation of the title, per the definition in Section 4. Over 95% of the poems submitted received a bonus (Table 5). Finally, for the judging task of comparing solo-writers and collaborative writers, crowdworkers have to read two poems and answer 3 questions, which takes on average 1 minute, so we set the payment to \$0.25 per HIT. All of these amounts were calculated according an hourly rate of 15\$ per hour.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. We additionally also want to acknowledge all human authors who posted their work open-sourced on the websites we collected the data from. Tuhin is funded by Columbia Center of Artifical Intelligence & Technology (CAIT) and the Amazon Science Ph.D. Fellowship. This work is also supported by the Samsung Advanced Institute of Technology (Next Generation Deep Learning: From Pattern Recognition to AI), the National Science Foundation under Grant No. 1922658, and a gift from AWS AI.

References

Nader Akoury, Shufan Wang, Josh Whiting, Stephen Hood, Nanyun Peng, and Mohit Iyyer. 2020. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6470–6484, Online. Association for Computational Linguistics.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo Ni, et al. 2021. Ext5: Towards extreme multi-

task scaling for transfer learning. *arXiv preprint arXiv:2111.10952*.

Gwern Branwen. 2020. Gpt-3 creative fiction.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval*, pages 806–810. Springer.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257 – 289.

Tuhin Chakrabarty, Arkadiy Saakyan, and Smaranda Muresan. 2021. Don't go far off: An empirical study on neural poetry translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7253–7265, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Quanze Chen, Chenyang Lei, Wei Xu, Ellie Pavlick, and Chris Callison-Burch. 2014. Poetry of the crowd: A human computation algorithm to convert prose into rhyming verse. In *HCOMP*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, page 329–340, New York, NY, USA. Association for Computing Machinery.

Andy Coenen, Luke Davis, Daphne Ippolito, Emily Reif, and Ann Yuan. 2021. Wordcraft: a human-ai collaborative editor for story writing. *arXiv preprint arXiv:2107.07430*.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501, Online. Association for Computational Linguistics.

Wanyu Du, Zae Myung Kim, Vipul Raheja, Dhruv Kumar, and Dongyeop Kang. 2022. Read, revise, repeat: A system demonstration for human-in-the-loop iterative text revision. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 96–108, Dublin, Ireland. Association for Computational Linguistics.

Katherine Elkins and Jon Chun. 2020. Can gpt-3 pass a writer's turing test? *Journal of Cultural Analytics*, 5(2):17212.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Austin, Texas. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*.

Arthur M Jacobs. 2018. The gutenberg english poetry corpus: exemplary quantitative narrative analyses. *Frontiers in Digital Humanities*, 5:5.

Mina Lee, Percy Liang, and Qian Yang. 2022. Coauthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA. Association for Computing Machinery.

Chin-yew Lin and Marina Rey. 2004. Looking for a few good metrics: ROUGE and its evaluation. In *NTCIR Workshop*.

Piotr Mirowski, Kory W Mathewson, Jaylen Pittman, and Richard Evans. 2022. Co-writing screenplays and theatre scripts with language models: An evaluation by industry professionals. *arXiv preprint arXiv:2209.14958*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

Swaroop Mishra and Elnaz Nouri. 2022. Help me think: A simple prompting strategy for non-experts to create customized content with models. *arXiv preprint arXiv:2208.08232*.

Aitor Ormazabal, Mikel Artetxe, Manex Agirrezabal, Aitor Soroa, and Eneko Agirre. 2022. Poelm: A meter-and rhyme-controllable language model for unsupervised poetry generation. *arXiv preprint arXiv:2205.12206*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Preprint*.

Vishakh Padmakumar and He He. 2022. Machine-in-the-loop rewriting for creative image captioning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 573–586, Seattle, United States. Association for Computational Linguistics.

Vishakh Padmakumar, Leonard Lausen, Miguel Ballesteros, Sheng Zha, He He, and George Karypis. 2022. Exploring the role of task transferability in large-scale multi-task learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2542–2550, Seattle, United States. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. 2022. A recipe for arbitrary text style transfer with large language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 837–848, Dublin, Ireland. Association for Computational Linguistics.

Melissa Roemmele and Andrew S. Gordon. 2015. Creative help: A story writing assistant. In *ICIDS*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv*.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019. Do massively pretrained language models make better storytellers? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 843–861, Hong Kong, China. Association for Computational Linguistics.

Ben Swanson, Kory Mathewson, Ben Pietrzak, Sherol Chen, and Monica Dinalescu. 2021. Story centaur: Large language model few shot learning as a creative writing tool. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 244–256.

Yufei Tian and Nanyun Peng. 2022. Zero-shot sonnet generation with discourse-level planning and aesthetics features. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3587–3597, Seattle, United States. Association for Computational Linguistics.

David Uthus, Maria Voitovich, and R.J. Mical. 2022. Augmenting poetry composition with Verse by Verse. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 18–26, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

David Uthus, Maria Voitovich, RJ Mical, and Ray Kurzweil. 2019. First steps towards collaborative poetry generation.

Tim Van de Cruys. 2020. Automatic poetry generation from prosaic text. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2471–2480.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *arXiv*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Kevin Yang, Nanyun Peng, Yuandong Tian, and Dan Klein. 2022. Re3: Generating longer stories with recursive reprompting and revision. *arXiv preprint arXiv:2210.06774*.

A Appendix

A.1 Creation of Instructions

To create instructions for a particular "Subject" we detect all possible noun phrases from an individual poetic sentence and create a natural language instruction for each of them using the template describe in Table 1. For "End" we fill the respective instruction template with the ending word in a sentence. For the"Rhyme" instruction we first find all rhyming words for the ending word in a sentence using the CMU Pronouncing Dictionary [12] and then fill the instruction template with a random rhyming word to ensure diversity. For the "Next Sentence" we fill the instruction template with its previous context sentence from any given poetry. To create "Metaphor" instruction we crawl websites for outputs of the form "NP1 is NP2" and fill NP1 in the template. A 'Simile" usually consists of two noun phrases typically a Subject and an Object with an usual syntax "NP1 is like NP2". We fill the Subject NP1 in the instruction template and manually edit it by expert humans for any inconsistencies. It should be noted that both output quality and instructions for Simile and Metaphors are manually inspected and agreed upon by two expert humans and only examples with full agreement are kept in the data. To create the instruction for 'Haiku' we need to fill the template with its title which is not always readily available. Hence we use YAKE (Campos et al., 2018, 2020), an unsupervised automatic keyword extraction method for selecting salient words from the Haiku that serves as its title. For Onomatopoeia we compile a lexicon containing words [13] representing them and then filter out sentences with any noun subject containing a word from the lexicon.

[12] https://pypi.org/project/pronouncing/

Instruction Type	Source	Stats
Lexical Constraint	Poetry Translation Corpus Chakrabarty et al. (2021)	94.5%
Continuation	Poetry Translation Corpus Chakrabarty et al. (2021)	3.18%
Rhetorical Devices	r/OCPoetry , r/Poetry Gutenberg Jacobs (2018), DMDMQ [14]	1.12%
Haiku	r/Haiku	1.14%

Table 7: Instruction Types along with the source from where the data is collected.

The AI is very competent and helpful,it's enjoyable to work with it.
I think it works very fine and I wish I had this whenever I had writer's block.
The best part of the tool is getting help with words at the end of a sentence and then being able to build off that.

Table 8: Some of the feedback from experts on the helpfulness of using our CoPoet system.

B Poems from User Study

We attach further examples of poetry written in collaboration with CoPoet in Figures 6 to 9. These include instances where the user selects none of the options presented to them (Figures 6 and 8) and highly intertwined collaboration where the user frequently rewrites model output (Figure 9). Additionally, Figure 10 is a word cloud of the titles of all the poems written by the users.

C User Interface

A snapshot of our interface during the user study can be found in Figure 11. The user is presented with a text box to edit their poem draft along with a dialog box to query the model. From an initial pilot, we observed that some users were not able to

[13] https://kathytemean.wordpress.com/2009/12/29/onomatopoeia-word-list/

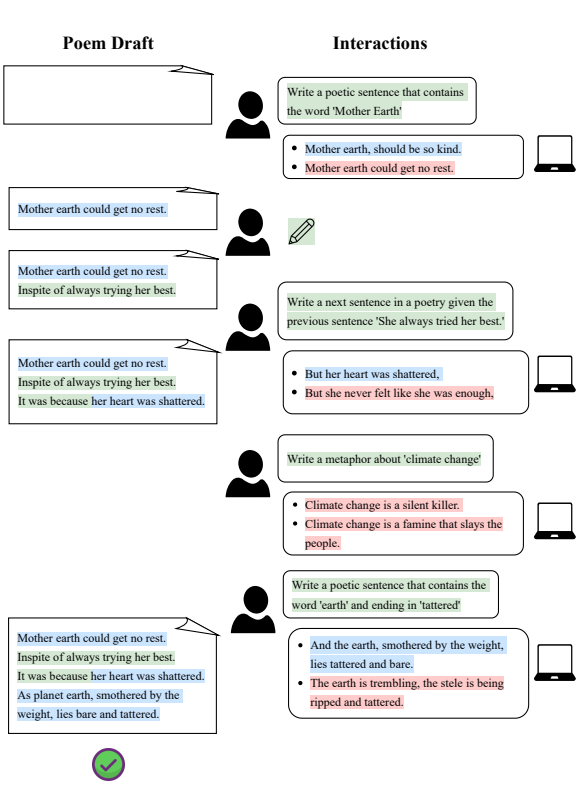

Figure 6: Poem entitled 'Climate Change' written in collaboration with CoPoet.

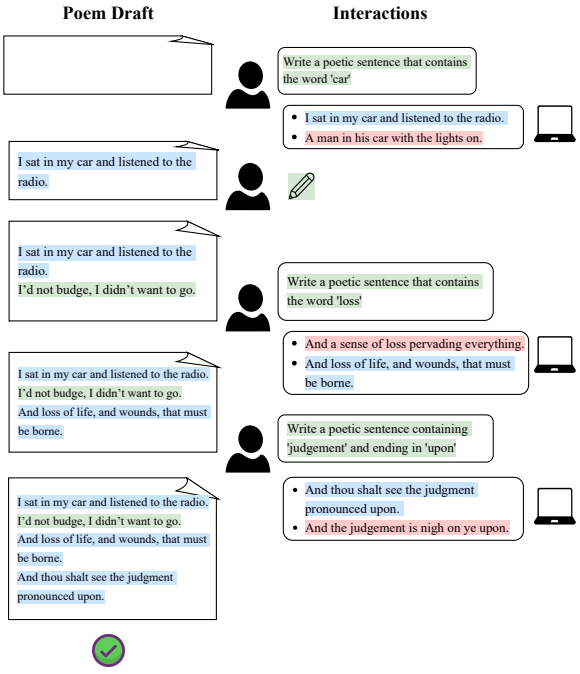

Figure 7: Poem entitled 'Courthouse Parking Lot' written in collaboration with CoPoet.

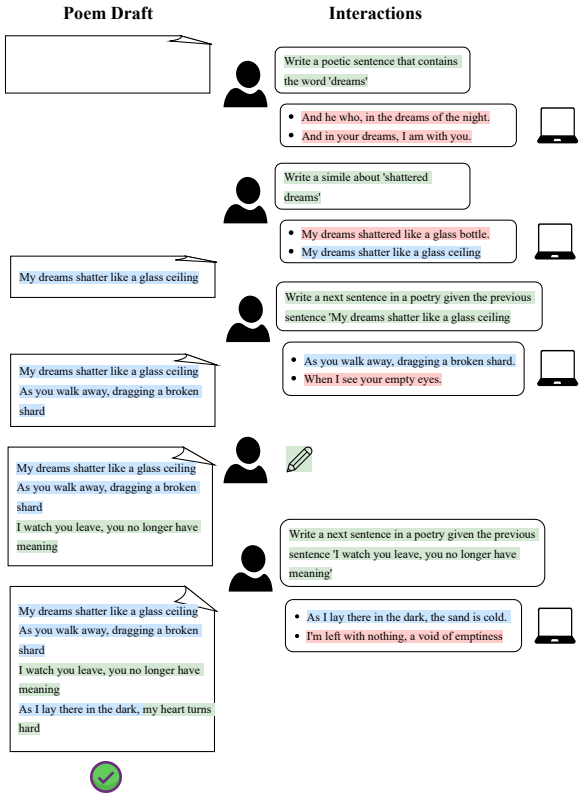

Figure 8: Poem entitled 'Glass Ceilings' written in collaboration with CoPoet.

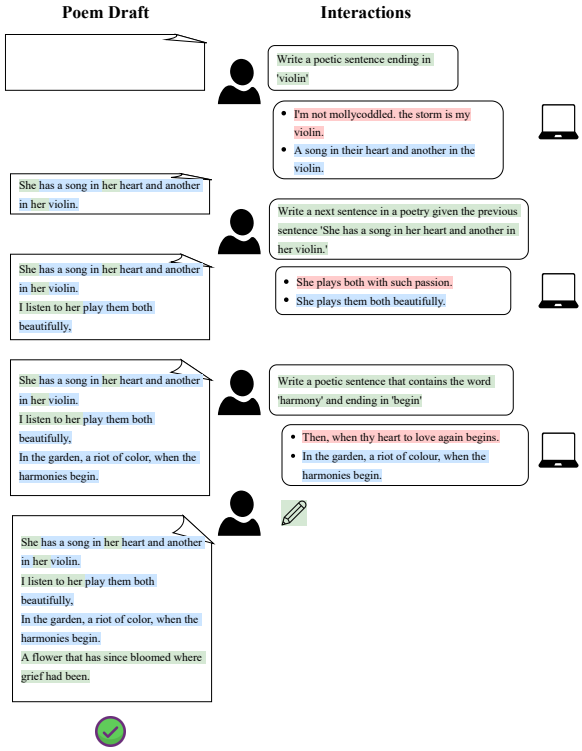

Figure 9: Poem entitled 'Petal Melody' written in collaboration with CoPoet.

Algorithm 1 Algorithm to compute how much of final submitted poem is written using model output.

Let $S = \{i, o\}$ be the set of all instructions requested by the experts and corresponding model outputs for a **single poem**;
$sum_RL = 0; n = num_lines(poem)$

for each line $l \in poem$ **do**
  $max_rouge = -1$
  $max_tuple = None$
  for each $(i', o') \in S$ **do**
    $rouge_score = RougeL(o', l)$
    if $rouge_score > max_rouge$: **then**
      $max_rouge = rouge_score$
      $max_tuple = (i', o')$
    end if
  end for
  $sum_RL = sum_RL + max_rouge$
  $S' = S$; $S' = S' - max_tuple$; $S = S'$
end for
$poem_RL = sum_RL/n$

effectively write instructions. As a result, we chose to provide instruction templates as part of the interface in the form of radio buttons (Figure 11). This was aimed at informing novice writers of the kind of instructions that elicit creative lines of text (rhymes, metaphors, etc.) from the model which they can then use to write better poems (which we noted in some feedback obtained from crowdworkers). However, this also primes the users to write instructions similar to the templates—almost all the instructions used by the crowdworkers belonged to the templates provided in the interface (or novel combinations of these). In this work, we did not perform an extensive comparison of different interface designs which could influence the effectiveness of human-AI collaboration. Our main goal is to design and test the instruction tuning setup specifically for the poetry writing task, which was why we chose to retain the interface design with the templates. The user interface ensures that most of the queries to the model follow the same templates which need not be the case in deployment. From very preliminary experiments, we see that InstructGPT3 outperforms our fine-tuned model on completely unseen instructions, and we intend to investigate this setting in detail going forward. We believe that as we provide users with greater flexibility in how to use the interface, the associated model must be able to respond robustly to the edge cases that users might provide and hence needs extensive rounds of piloting prior to deployment.

Figure 10: Word cloud of different concepts from titles of Poetry.

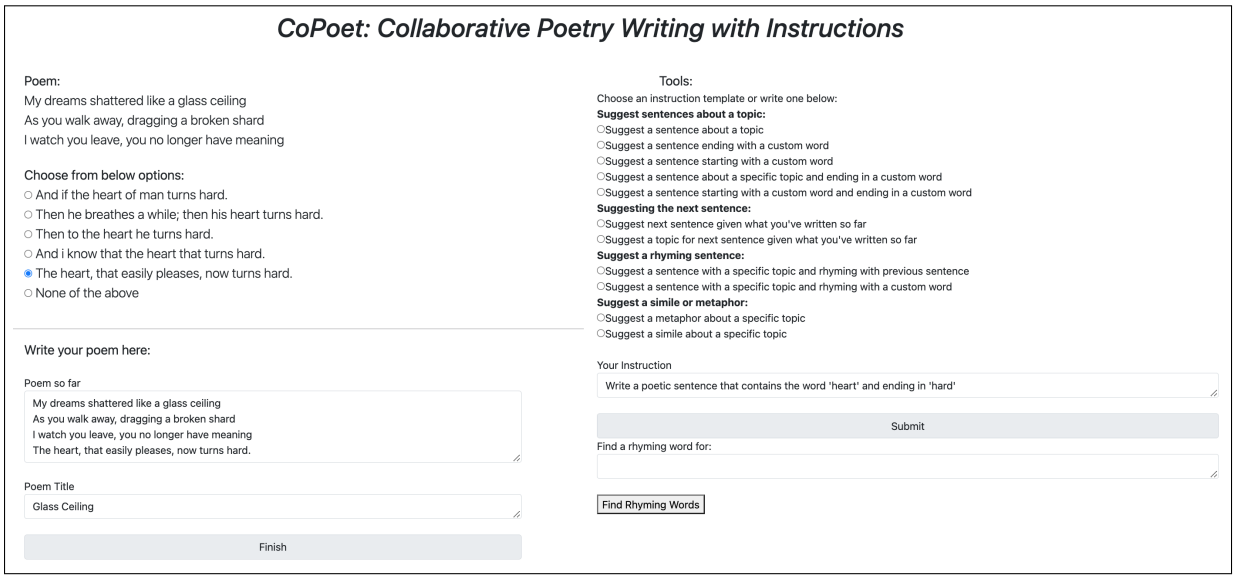

Figure 11: Snapshot of CoPoet: Collaborative Poetry Writing with Instructions